多次元状態-行動空間での強化学習

Reinforcement Learning in High-dimensional State-Action Space

九州大学 大学院工学研究院 海洋システム工学部門　○木村 元

○ Hajime Kimura, Graduate School of Engineering, Kyushu University

Abstract: In real-robot applications, learning controllers are often required to obtain control rules over high-dimensional continuous state-action space. Random coarse coding is a promising method to deal with high-dimensional state space for representing the state-value function. However, there is no standard reinforcement learning scheme to deal with action selection in high-dimensional action space, especially the probability of action variables are mutually dependent. This paper introduces a new action selection scheme using random-rectangular coarse coding and Gibbs sampling, and shows Q-learning for the proposed scheme. The proposed approach is demonstrated through some simulations of robots.

1　はじめに

本論文では，ランダム矩形タイルを用いて状態-行動空間を汎化し，Q 値や行動選択確率分布を表現する方法を紹介する。次に，多次元で連続な行動空間を細かく離散化するが，Gibbs サンプリングによって空間爆発を回避しつつ行動選択を行う方法を提案する。Q-learning は代表的強化学習手法であるにもかかわらず，行動選択の困難さから高次元の行動空間を持つロボットへ実装することが困難だったが，本手法により容易に実装できることを示す。

2　連続空間での行動価値関数表現

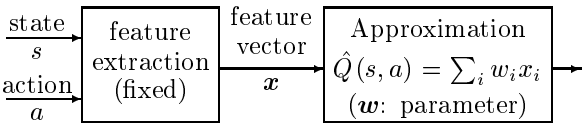

Figure 1: Linear architecture for generalization of Q-function

Q-learning に代表される強化学習アルゴリズムの基本は，状態入力から状態・行動評価値への写像関数を構築していくことである。状態表現の生成は特徴抽出に関連するが，強化学習ではこの部分の学習は行わず，予め与えられたものを使うが一般的である。状態入力が連続空間の場合，このテクニックは汎化 (generalization) や関数近似と呼ばれる。Q-learning の最適解への収束が保障される汎化法として線形アーキテクチャがある。これは固定された基底関数の集合を用いて関数近似を行うもので，ラジアル基底やシグモイド関数を用いたものなど様々だが，状態-行動入力を基底関数によって特徴ベクトル $\boldsymbol{x}$ に変換し，このベクトルの線形重み付け和で Q 値を表現する点で共通である (Fig.1)。特徴ベクトルの生成方法は問題に応じて設計者が与える。離散空間 Q-learning との整合性から，特徴ベクトルのノルムは常に 1 とするのが望ましい。

Fig.2 は最も単純な特徴ベクトル生成方法の例を示す。２次元の入力空間を１種類のグリッドタイリングによって分割し，各タイル要素に特徴量ベクトルの要素を割り当てる。状態-行動入力があるタイル要素内にあるとき，該当する特徴量ベクトル要素の値は 1 に，それ以外の要素はゼロとする。Q 値は，特徴ベクトル要素と同数の重み変数 w_i

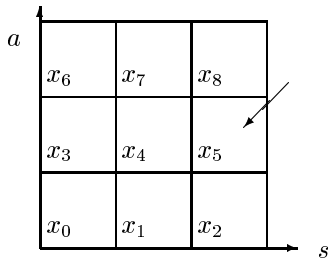

Figure 2: An example of tile coding to generate a feature vector. When the current state (s_1, s_2) is located at the arrow in the figure, the corresponding element x_5 equals 1, and the others are all zero in the feature vector $(x_0, x_1, \cdots x_8)$.

を用意し，特徴ベクトルとの線形和で表す：

$$\hat{Q}(s,a) = \sum_{i=1}^{n} w_i x_i \tag{1}$$

ただし n は特徴ベクトル要素の個数である。Q 値を変える場合は，重み変数 w_i の値を調節する。線形アーキテクチャによる Q-learning は，単純な勾配法に基づき重み変数 w_i を更新することで達成される：

$$w_i \leftarrow w_i + \alpha \frac{\partial \hat{Q}(s,a)}{\partial w_i} \delta_t \tag{2}$$

式 (1) より $\partial \hat{Q}(s,a)/\partial w_i = x_i$ なので，

$$w_i \ \leftarrow \ w_i + \alpha \ x_i \ \delta_t \tag{3}$$

つまり線形アーキテクチャを用いた更新では，更新する重み変数 w_i に対応する特徴の要素の値 x_i だけを使って簡単に更新できる。Fig.2 のような単純なグリッド分割による線形アーキテクチャの更新式は，離散 Q-learning の更新式と同一である。一般に，全ての状態-行動ペアが互いに線形独立な特徴ベクトルで表現されていれば最適行動価値関数への収束が保障される [2]。Fig.3 に線形アーキテクチャを用いた Q-learning の処理手順をまとめる。

2.1　高次元空間での特徴ベクトル生成

前節で説明した単純なグリッドタイリングやこれに準ずる方法では，高次元空間においてメモリ爆発の問題を引き起

1. エージェントは環境の状態 s_t を観測する。
2. エージェントは任意の行動選択方法 (探査戦略) に従って行動 a_t を実行する。
3. 環境から報酬 r_t を受け取る。
4. 状態遷移後の状態 s_{t+1} を観測する。
5. 状態 s，行動 a に対する特徴ベクトル $\boldsymbol{x}=(x_1, x_2, \cdots x_n)$ を何らかの方法を用いて生成する。ただし，区別が必要な状態-行動ペア間では必ず互いに線形独立になるよう設定し，ノルムを 1 にしておく。
6. 特徴ベクトルの要素数と同数の重み変数 w_i を用いて Q 値を計算：
$$\hat{Q}(s,a) = \sum_{i=1}^{n} w_i x_i \qquad (4)$$
7. 以下の更新式により重み変数 w_i を更新する：
$$\delta_t = r_t + \gamma \max_a \hat{Q}(s_{t+1}, a) - \hat{Q}(s_t, a_t)$$
$$w_i \leftarrow w_i + \alpha\, x_i\, \delta_t$$
ただし α は学習率，γ は割引率。
8. 時間ステップ t を $t+1$ へ進めて手順 1 へ戻る。

Figure 3: Q-learning scheme using the linear architecture

こす。そこで，ランダム矩形タイルを用いた特徴ベクトル生成方法を紹介する。ある 1 つのランダムタイル $f(x)$ は，入力空間 x を構成する次元のうち，任意の 1 次元以上の複数次元（sensitive element と呼ぶ）で定義される部分空間中の，ある矩形領域を表し，入力された座標 x がその矩形領域である場合 $f(x)=1$ を出力し，それ以外の場合 $f(x)=0$ である。矩形領域を構成する次元や矩形の範囲・大きさなどは，タイル毎にランダムに与えるが，その与え方には統計的性質を持たせる[1] 。このように，ある 1 つのタイルは空間を構成する次元の一部分で構成される空間の，とある領域として処理するが，このタイルを空間全体からみると，定義される部分空間中では範囲の限定された矩形だが，それ以外の空間では全領域をカバーするタイルと同義である。Fig.4 に 2 次元での例を示す。ランダムなタイルを用いることに抵抗を覚える読者が多いと思うが，実験してみると驚くべきことに同数の特徴量を持つ等間隔の格子状グリッドゴーディングに比べ，比較にならないほど高性能である。

3 高次元行動空間での行動選択

前節の汎化方法によって高次元空間において複雑な状態-行動評価関数（Q 関数）が表現できるが，この Q 関数に従って行動を選択する最も単純な方法は，フラットな行動選択法である。これは，行動空間を全て細かい格子状に離散化し，各超矩形領域に対して確率を割当て，ルーレット選択によっていずれかの超矩形領域に相当する行動を選択するものである。本稿では，連続な行動空間を各次元毎に等しく D 分割して格子状に離散化する。よって行動空間を N 次元と仮定すると D^N コの行動 a_i $(i=\{1,\cdots,D^N\})$ へ離散化される。行動は，各行動 a_i 毎に定義された確率

[1] 一様分布で入力が与えられたとき，ランダムタイルのヒット確率の期待値がある一定値（10 %程度）になるよう決める。

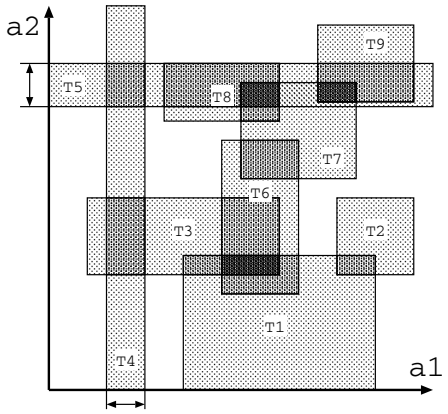

Figure 4: An example of the random rectangular coarse coding using 9 features in two-dimensional action space. The coarse feature T4 is sensitive to the component a1, and it is defined by the arrow width in the sub-space a1. But the T4 ignores the component a2, then the rectangle covers the other all sub-space a2. The feature T5 is defined similarly.

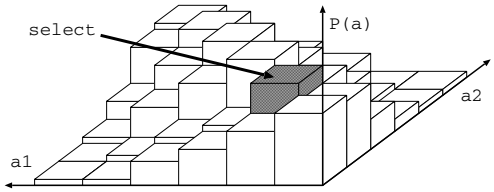

Figure 5: An action-selection with a finite-grid approach in two-dimensional action space. The probability of the action selection $P(a)$ is represented by quantizing the action space $a1-a2$ into a finite number of cells.

$P(a_i)$ に従って，どれか 1 つが選ばれる。Fig.5 は 2 次元の行動空間において 2 次元行動空間を格子状に分割し，各メッシュの持つ確率に応じて行動選択する様子を示す。フラットな行動選択では，2 次元空間 $a1-a2$ で離散化された全ての格子について確率を計算し，その比率に基づいて格子を一つ選択して行動出力とする。行動の確率 $P(a_i)$ は，一般に状態 s において行動 a_i に割り当てられた Q 値 $Q(s,a_i)$ に応じた以下のボルツマン分布とする：

$$P(a_i) = \frac{\exp\left(\frac{Q(s,a_i)-offset(s)}{T\times width(s)}\right)}{\sum_{j=1}^{D^N}\exp\left(\frac{Q(s,a_j)-offset(s)}{T\times width(s)}\right)} \qquad (5)$$

一般的なボルツマン行動選択では $offset(s)=0$ かつ $width(s)=1$ であるが，行動空間の次元数が高い場合 $offset(s)$ を状態 s における Q 値の平均値，$width(s)$ を状態 s における Q 値の標準偏差とすることを推奨する[2] 。これは Q 値の大きさによって行動選択の挙動が温度パラメータ T を無視して大きく変化してしまうのを防ぐためである。しかしながら式 (5) で明らかなように，フラットな行動選択では全行動空間の Q 値を調べる必要があるため，行動次元数が 2〜3 次元程度ならば実行可能だが，次元数がさらに高

[2] Q 値の平均値や標準偏差は，状態 s において行動 a を Q 値に関係なく一様分布で 10〜20 個選んだときの各行動の Q 値から統計的に計算した値で良い。

くなると離散化した行動空間が指数的に増大するため，記憶容量的にも速度的にも実行不能になる。

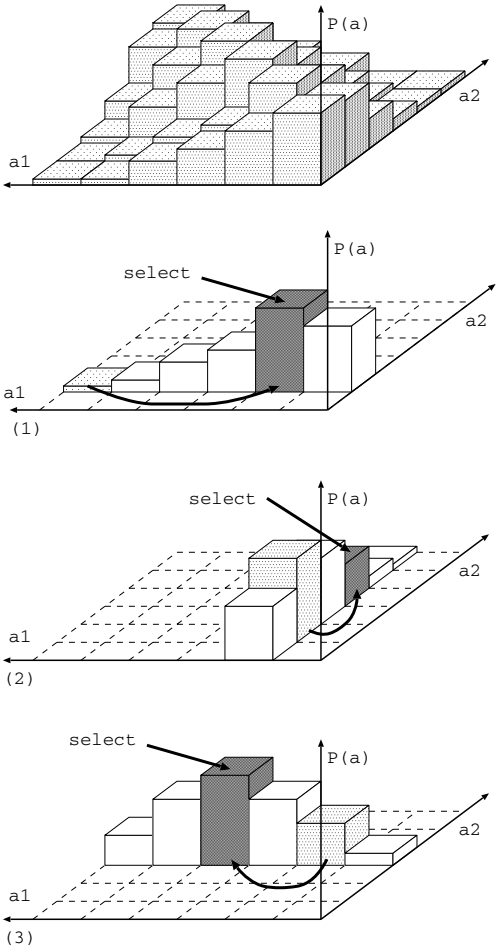

Figure 6: An example of the Gibbs-sampling scheme with the finite-grid approach in two-dimensional action space. The top left shows a joint distribution of the action selection probability. The top right (1) represents selecting action $a1$ following the conditional probability given $a2$. The bottom left (2) represents selecting action $a2$ following the conditional probability given $a1$ at the top right (1). The bottom right (3) represents selecting action $a1$ again, following the conditional probability given $a2$ at the bottom left (2).

そこで，高次元空間における確率分布に従って効率良くサンプリングする方法として Gibbs サンプリングを用いる行動選択法を提案する。Gibbs サンプリングとは，高次元の確率変数において，注目している次元以外の次元の変数の値を固定し，そのときの条件付確率分布を用いて 1 次元ずつ順番にサンプルを行っていく処理を全ての次元に対して十分な回数繰返し，最終的に得た値をサンプルとする Markov chain Monte-Carlo 法（MCMC 法）の一種である。Fig.6 は 2 次元の行動空間における Gibbs サンプリングの様子を示す。2 次元空間 $a1-a2$ で離散化された全ての格子における確率は Fig.5 のフラット選択の場合と同じだが，1 次元ずつサンプルを行っていくので，Fig.5 のような全行動空間における確率を全て計算するような処理が不要となる。Fig.6 の (1) は $a2$ を固定した条件付確率により $a1$ をサンプルし，その下 (2) では (1) で選んだ $a1$ を固定した条件付確率により $a2$ をサンプルする。これで反復 1 回分であり，さらに (3) では (2) で選んだ $a2$ を固定した条件付確率により再び $a1$ を選ぶ。このような反復を十分な回数繰返した結果得られた $a1, a2$ を最終的な行動出力とする。Gibbs サンプリングでは，反復回数についての理論的な下限については現在のところ明らかにはなっておらず，実験的に決められているのが実情である。Fig.6 で示した例題のように 2 次元程度の空間での 6×6 程度の粗い分割では，フラットな行動選択との計算量的な差はあまりないが，行動次元数や分割数が増加すると，その差は顕著に現れる。例えば行動空間が 8 次元で各次元を 10 分割する場合，フラットな行動選択では各メッシュにおける重みの計算を 10^8 =1 千万回行わなければならないが，Gibbs サンプリングでは $10 \times 8 \times$（反復回数）程度であり，反復回数を数十回程度に抑えればフラットな行動選択法に比べ，1 万分の 1 程度の計算量で済む。

一般的な Gibbs サンプリングでは行動選択確率分布自体は式 (5) と同じだが，各行動座標軸毎に処理を行うため，記法を以下のように整理する。フラットな行動選択と同様，行動の次元数 N で，行動空間は各次元毎に D 分割により離散化されているものとする。ある多次元行動 a について，各行動次元毎に分解して次のように表す：$a = (a^1, a^2, \ldots a^N)$ ただし各次元の要素 a^n (ただし $n \in \{1, 2, \cdots N\}$) は $a^n \in a^n_d$ ただし $d \in \{1, 2, \cdots D\}$ である。ある状態 s 行動 a に対する Q 値は，$Q(s, a) = Q(a^1, a^2, \ldots a^N | s)$ と表す。この Q 値は，フラット選択における値と同一である。Gibbs サンプリングの t 回目の反復においてサンプルされた行動要素を $a^1(t), a^2(t), \ldots a^N(t)$ と表す。このとき，$t+1$ 回目の反復で行動要素は以下の確率に従ってサンプルされる：

$$\begin{aligned} a^1(t+1) &\sim P(a^1 | a^2(t), a^3(t), \ldots a^N(t)) \\ a^2(t+1) &\sim P(a^2 | a^1(t), a^3(t), \ldots a^N(t)) \\ &\vdots \\ a^N(t+1) &\sim P(a^N | a^1(t), a^2(t), \ldots a^{N-1}(t)) \end{aligned}$$

ここで，条件付き確率 $P(a^n_i | a^1, a^2, \ldots a^N)$ は，以下のボルツマン分布で与えられる：

$$P(a^n_i | a^1, a^2, \ldots a^N) = \frac{\exp\left(\frac{Q(a^1, a^2, \ldots a^n_i, \ldots a^N | s) - offset(s)}{T \times width(s)}\right)}{\sum_{d=1}^{D} \exp\left(\frac{Q(a^1_d, a^2_d, \ldots a^n_d, \ldots a^N_d | s) - offset(s)}{T \times width(s)}\right)}$$

フラットな行動選択と比べると，考慮すべき行動が 1 次元ずつになり，式 (5) と同じような処理を繰り返し行うようになっただけである。このとき，全行動空間について Q 値を調べる必要がないのが大きな利点である。

この Gibbs サンプリングによる行動選択は，Q-learning において $\max_a Q(s', a)$ を探す場合，Gibbs サンプリングを行う過程で見つかった最大の Q 値を用いるなど近似的に与えることになるが，前節で説明したランダムタイリングと組合せると，3〜8 次元を超える高次元の行動空間を持つ強化学習問題に対して非常に効果的である。

4 実験

Fig.7 に示す冗長アームのリーチングタスクに提案手法を適用した。アームは 8 本のリンクが関節を介して直列に接

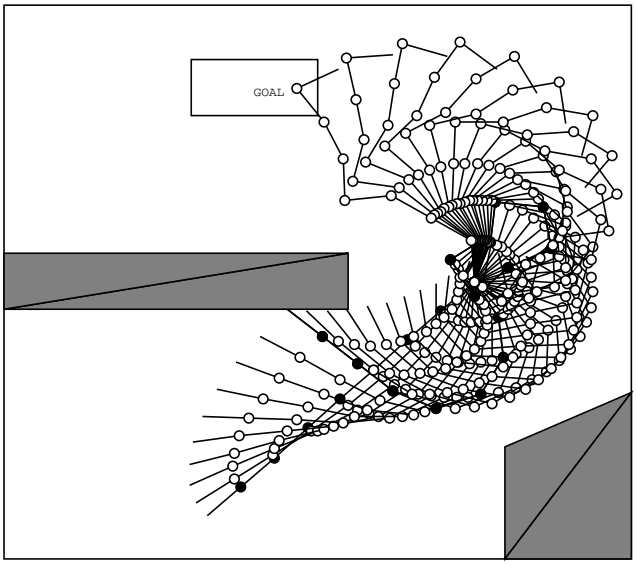

Figure 7: A learned example behavior of a reaching task using redundant-arm that have eight-joints. The conditions that the joints are shown by black circles are the positions at which the learner is making decisions.

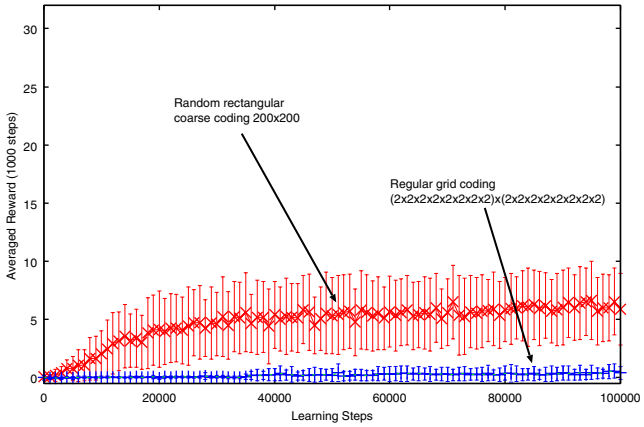

Figure 8: Learning results averaged over 10 trials in the 8-joints redundant-arm reaching task. The vertical axis gives the averaged reward over making 1000 decision steps.

続されており，2 次元平面上を動く。行動も 8 次元で，目標とする関節角度を表す。行動により目標角度が与えられると，その方向へ 8 次元状態空間中を直線的に移動し，途中で障害物あるいはゴール領域に触れるか，目標座標に到達すると，イベントが発生して次の意思決定を行う。問題設定の詳細は文献 [4] を参照のこと。

強化学習エージェントは，8 次元の状態入力の各次元を 10 分割して離散化する。8 次元の行動についても同様に各次元を 10 分割して離散化する。この空間に対し，ランダムタイルを 200 個生成して特徴ベクトルとする。ランダムタイルは，一様分布で入力が与えられたとき、タイルのヒット確率が 0.1 になるように sensitive element の次元とタイルの幅に関する統計量を計算し，これに従って確率的に生成する。Gibbs サンプリングの反復回数は 40 回，割引率 $\gamma = 0.9$，温度 $T = 0.4$，学習率 $\alpha = 0.2$ である。参考までに，状態と行動空間を $2^8 = 256$ 個のタイルへ均一にグリッド分割した場合と比較した。また、フラット行動選択を行った場合との行動選択時間の比較を試みたが、フラット行動選択におけるボルツマン分布の分母の値が大きくなり過ぎて計算に支障をきたすなど実装上の問題が発生し，実行不能であった。Fig.8 に学習の様子（10 試行平均）を示す。

実験結果より，タスクの学習に必要なタイル数（特徴量）については，高い次元の問題が，多くのタイルを必要とするとは限らないことが考察される。本問題では，アームの移動コストを考えず，意思決定ステップ数だけで評価しているため，最適なパスが無数に存在するために，200 個という比較的少ないタイルでも最適解を容易に発見できたと考えられる。しかし，同じような分割数・タイル数でも均一にグリッド分割した場合には全く学習できなかった。ランダムタイリングにおける細かい分割と粗い分割の混合や，全ての次元の変数を考慮するのではなく一部の次元の変数に注目した汎化が重要と考えられる。線形関数近似を用いた Value 関数の強化学習では，学習が収束するための条件の一つとして特徴ベクトルが状態間で線形独立であることはすでに述べた。一般的な格子分割による離散化や，ラジアル基底関数による関数近似は，ほぼ全ての領域で単位ベクトルに近い線形独立な特徴ベクトルが生成されることが保障されるが，特徴ベクトルが単位ベクトルであることは収束の条件としては必要ではない。また，本タスクをみれば明らかなように，連続な状態-行動空間におけるタスクは，本質的には複雑さが小さく，区別が必要な状態や行動の領域はあまり多くないと考えられる。提案手法であるランダムな矩形タイルによる特徴ベクトル生成法は，上記のような本質的に複雑さの小さい連続な状態-行動空間におけるタスクにおいて，区別が必要な状態や行動の領域において線形独立な特徴量ベクトルを生成しやすいと考えられる。つまり，提案手法では特徴量ベクトルが単位ベクトルに制限されていないため，タスクに無関係にランダムに配置されたタイルでも，線形独立な特徴量ベクトルになりやすく，適切な関数近似が行えるのではないかと考えられる。また、行動選択については，離散 Q-learning において一般的なボルツマン行動選択を Gibbs-sampling というテクニックによって高次元行動空間での実行を可能とした。さらに，ランダムな矩形タイルによる特徴ベクトル生成法は，Gibbs-sampling において計算処理を簡略化できるため，これらの組合せは効果的である。

References

[1] Watkins, C.J.C.H. & Dayan, P.: Technical Note: *Q*-Learning, Machine Learning 8, pp.279–292 (1992).

[2] Bertsekas, D.P. and Tsitsiklis, J.N.: ”Neuro-Dynamic Programming”, Athena Scientific (1996).

[3] Sutton, R. S. and Barto, A.: ”Reinforcement Learning: An Introduction”, A Bradford Book, The MIT Press (1998).

[4] 木村 元：ランダムタイリングと Gibbs-sampling を用いた多次元状態-行動空間における強化学習，計測自動制御学会論文集, Vol.42, no.12, pp.1336–1343 (2006).